RedgeX: Resource-Tailored Intelligence at the Edge

Meta-learning-based analytical-model generation for programmable edge devices

How can one analytical task adapt to edge devices with very different resources?

1 INDUSTRIAL GATEWAY A — CPU 25% — RAM 512 MB — STORAGE 8 GB

2 SMART CAMERA EDGE DEVICE — CPU 60% — RAM 1 GB — STORAGE 16 GB

3 SENSOR NODE (MOTION) — CPU 15% — RAM 256 MB — STORAGE 4 GB

4 MICRO EDGE COMPUTER — CPU 80% — RAM 4 GB — STORAGE 64 GB

5 MINI EDGE SERVER — CPU 35% — RAM 2 GB — STORAGE 128 GB

RESOURCE TELEMETRY (CPU, RAM, STORAGE)

K Original Code Task $\{S_k, C_k\}$

S_k: task size
C_k: CPU cycles

SDN-enabled RedgeX Controller

GLOBAL RESOURCE DASHBOARD

CPU (Avg.) 43% | RAM (Total) 7.8 GB / 16 GB | STORAGE (Total) 220 GB / 320 GB

META-LEARNING / MODEL-SELECTION ENGINE

Task Features → Meta-learning Model → Performance Data

Generates resource-tailored analytical model

K′ Resource-Tailored Code (Optimized) Task $\{S_k, C_k\}$

DEPLOYMENT (Install K′)

TARGET EDGE DEVICE

3 SENSOR NODE (MOTION)

Executes analytical task using *K′* $\{S_k, C_k\}$

REMOTE CLOUD

LEGEND
- Telemetry / Data Flow
- Meta-learning / Modeling
- Deployment / Execution
- Warning (High Usage)
- Inactive / Low Usage

Original task K $\{S_k, C_k\}$ → RedgeX optimization → Resource-tailored task K′ $\{S_k, C_k\}$ → Target edge device executes task

HIGHLIGHTS OF THE PAPER

01 META-LEARNING MODEL GENERATION

RedgeX uses task features, edge-resource states and performance data to generate a suitable analytical model.

02 GLOBAL-STATUS EDGE CONTROL

An SDN-enabled controller maintains an updated view of CPU, RAM and storage across edge devices.

03 RESOURCE-AWARE CODE DEPLOYMENT

Original code is installed when resources are sufficient; optimized code *K′* is generated and deployed when resources are constrained.

04 EXPERIMENTAL FINDINGS

RedgeX reduces task-execution latency by 20–27%. The reported average gap between original and optimized code metrics is ***60.6%*** *with Linear Regression and* ***61.1%*** *with k-NN.*

Evaluated using **1,000** tuples, five edge devices and three Industrial IoT tasks.

Adapt the analytical model to the edge device instead of forcing every device to run the same code.

IEEE WCNC 2024
DOI: 10.1109/WCNC57260.2024.10570981

IEEE

Why One Analytical Model Does Not Fit Every Edge Device

Different devices have different processing, memory and storage limits

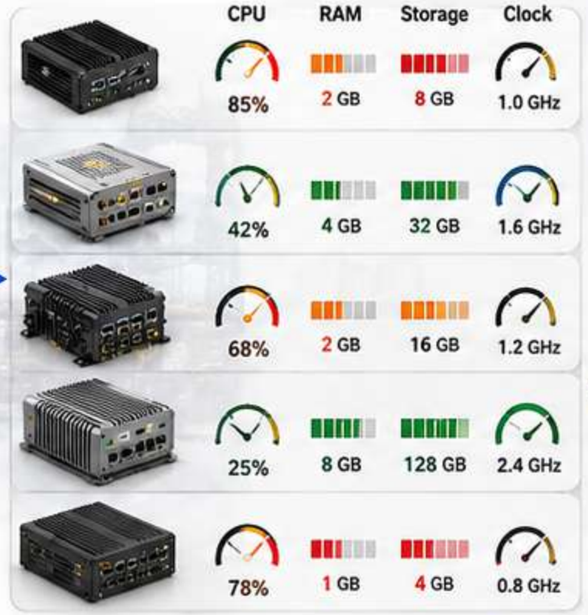
1 Industrial Task

$\{S_k, C_k\}$

S_k: program + input data size

C_k: required CPU cycles

2 Heterogeneous Edge Devices



?

Will the same code run efficiently on every device?

3 Original Code Sent Unchanged

- Resource Overload
- High Latency
- Wasted Capacity
- Failed Installation
- Manual Reconfiguration

4 Research Gap

Existing work mainly optimizes task offloading.

It does not select code snippets according to the current resource state of the target edge device.

$$C_k = S_k \times \rho$$

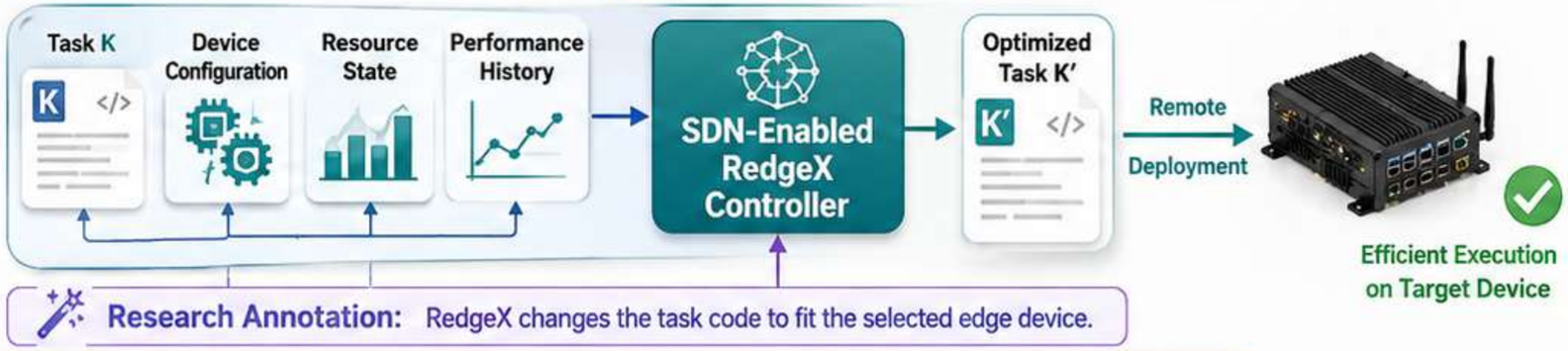
$$T_{k,n} = \frac{S_k \times \rho}{f_n}$$

Larger tasks and slower edge CPUs require more execution time.

Research Annotation:

Fixed analytical models and manual reprogramming do not scale across heterogeneous edge networks.

5 RedgeX Idea



Remote Deployment

Efficient Execution on Target Device

Research Annotation: RedgeX changes the task code to fit the selected edge device.

The problem is not only where to run a task, but also which version of the task should run there.

Inside the RedgeX Architecture

Global resource awareness and programmable task deployment

1. End device generates task $\{S_k, C_k\}$.
2. Task reaches nearest Primary Edge Node.
3. Primary node checks requirements and resources.
4. If resources are sufficient, execute original task K.
5. If resources are insufficient, send task and configuration to RedgeX Controller.
6. Controller checks global resource-status table.
7. Controller selects suitable target edge device.
8. Meta-learning creates optimized task K′ when required.
9. Controller remotely installs K or K′.
10. Edge device executes task and reports updated status.

TOP LAYER — CONTROL AND CLOUD

SDN

SDN-enabled RedgeX Controller

Global Resource Dashboard

CPU	RAM	Storage
72%	61%	58%

MIDDLE LAYER — COOPERATIVE EDGE NODES

	E_1	E_2	E_3	...	E_n
CPU	65%	48%	81%	55%	67%
RAM	58%	72%	64%	49%	66%
Storage	70%	59%	73%	66%	61%
Task Status	Idle	Busy	Idle	Busy	Idle
Perf. (ms)	18	24	16	22	19

BOTTOM LAYER — END DEVICES

Motion Sensor · Ultrasonic Sensor · LED Alert System · Industrial Camera · Robotic System · IoT Sensors / Actuators

$\{S_k, C_k\}$

Global State

The controller periodically collects resource information from all edge devices to support deployment and offloading decisions.

Main Resource State

CPU · RAM · Storage

Execution Decision

Resources sufficient? Yes → Install K (K)

No → Optimize K → Generate K′ (K′) → Deploy K′ (K′)

RedgeX combines edge-node selection with *task-code adaptation and remote reprogramming.* K / K′

From Original Code to a Resource-Tailored Edge Task

Meta-learning selects how the analytical task should be optimized

1 TRAINING DATA

Edge-device configurations
e.g., core count, architecture

Task code snippets
e.g., C/C++ code fragments

CPU availability
e.g., 2, 4, 8 cores

RAM availability
e.g., 2 GB, 4 GB, 8 GB

Storage availability
e.g., 8 GB, 16 GB, 32 GB

Measured performance
e.g., latency, throughput

Task type
e.g., RoboMovement, RoboDistance, RoboAlert

2 DATA PREPARATION

Data Collection
Gather device states, tasks and performance measurements

Data Cleaning
Remove noise, fix missing or inconsistent data

Data Selection
Select relevant records for learning

Feature Selection
Choose informative features that impact performance

3 META-LEARNING

Learn which optimization works for each task + device state

Candidate optimization models

Apply models to training tasks

Observe performance
e.g., latency, throughput, resource usage

Validation selects the best model / strategy

4 CODE OPTIMIZATION

Original task code: ***K***

```
1 for (i = 0; i < N; i++) {
2   for (j = 0; j < M; j++) {
3     A[i][j] = A[i][j] + B[i][j];
4     }
. }
```

Loop Unrolling
Reduce repeated loop overhead

Cache Blocking
Improve data locality

Parallelism
Use multiple processing units

Tools may be combined depending on available resources

5 OPTIMIZED TASK

Original task ***K***

- Higher resource demand (e.g., CPU, RAM, storage)
- Slower execution
- Poor fit for current resources

Optimized task ***K′***

- Fits available resources
- Faster execution
- Better hardware match
- Remotely deployable

K → ***K′***

6 DEPLOYMENT DECISION

Are available resources sufficient for ***K***?

Yes → Install original task ***K*** on target edge device

No → Optimize code → Generate ***K′***

Does ***K′*** fit available resources?

Yes → Install ***K′*** on target edge device

No → Revise optimization or select another suitable device

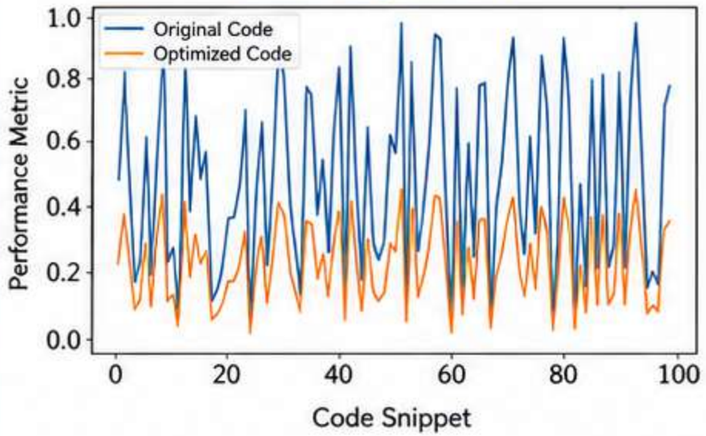
RESEARCH ANNOTATION: The meta-learning system learns which code-optimization choices work best for different task and device configurations.

RedgeX learns how to tailor the analytical code before deploying it to the edge.

What the RedgeX Experiments Show

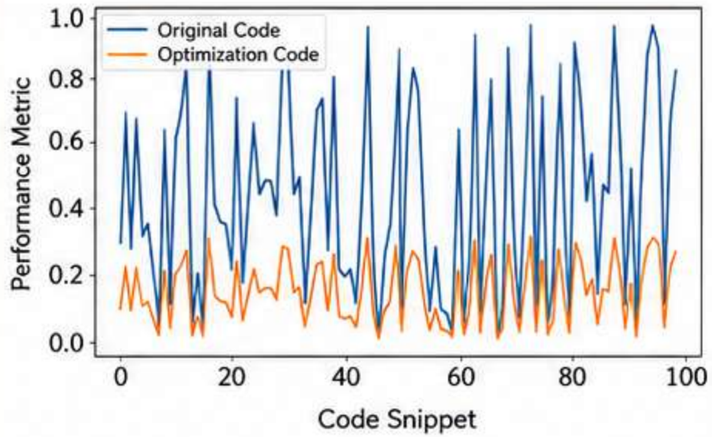
Original code versus resource-optimized edge tasks

1 Original vs Optimized Code — Linear Regression



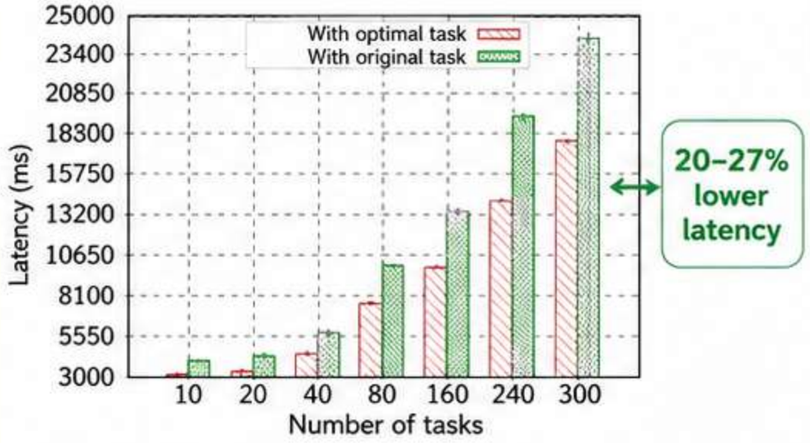
Average reported gap: **60.6%** between original and optimized task-code metrics.

2 Original vs Optimized Code — k-NN



Average reported difference: **61.1%** between original and optimized task-code metrics.

3 Task-Execution Latency



Optimized tasks reduce execution latency as the number of tasks increases.

Experimental Setup: 1,000 data tuples | 5 edge devices | 100 monitored configurations | RoboMovement | RoboDistance | RoboAlert | Linear Regression | k-NN | REdge experimental environment

Global resource state → select suitable edge device → optimize code snippets → generate *K′* → remote deployment → lower execution latency

Current Evaluation Scope

The evaluation uses five edge devices and three IoT tasks. Real-life IoT edge deployment is future work.

Smart Homes | Industrial IoT | Autonomous Vehicles

RedgeX dynamically generates resource-tailored analytical tasks, improving edge-resource utilization and reducing task-execution latency.

Future work: Deploy and evaluate RedgeX in a real-life IoT edge environment.

Observe the resources. Optimize the code. Deploy the right analytical task.